



Atomic Energy
Control Board

P.O. Box 1046
Ottawa, Canada
K1P 5S9

Commission de contrôle
de l'énergie atomique

C.P. 1046
Ottawa, Canada
K1P 5S9

INFO-0247

COMPUTER SOFTWARE
CONFIGURATION MANAGEMENT

by

G. Pelletier
PRIOR Data Sciences Ltd.

A research report prepared for the
Atomic Energy Control Board
Ottawa, Canada

Project no. 2.109.1

August 1987

COMPUTER SOFTWARE CONFIGURATION MANAGEMENT

ABSTRACT

This report reviews the basic elements of software configuration management (SCM) as defined by military and industry standards. Several software configuration management standards are evaluated given the requirements of the nuclear industry. A survey is included of available automated tools for supporting SCM activities. Some information is given on the experience of establishing and using SCM plans of other organizations that manage critical software. The report concludes with recommendations of practices that would be most appropriate for the nuclear power industry in Canada.

RÉSUMÉ

Le présent rapport examine les éléments fondamentaux de la gestion de la configuration des logiciels selon la définition des normes militaires et industrielles. Il évalue plusieurs normes de gestion de la configuration des logiciels à la lumière des exigences du secteur industriel nucléaire. Il comprend une étude sur les outils automatisés qui sont disponibles pour assurer les activités d'une telle gestion. Il fournit aussi des renseignements sur l'expérience acquise par certains autres organismes lorsqu'ils ont établi leur plan de gestion de la configuration des logiciels critiques. Il conclut enfin en recommandant l'établissement de certaines pratiques très appropriées pour le secteur industriel électronucléaire au Canada.

Disclaimer

The Atomic Energy Control Board is not responsible for the accuracy of the statements made or opinions expressed in this publication and neither the Board nor the author assume liability with respect to any damage or loss incurred as a result of the use made of the information contained in this publication.

10 August 1987

TABLE OF CONTENTS

ABSTRACT

1.	INTRODUCTION	1
2.	MANAGING SOFTWARE	2
2.1	Configuration Management	2
2.2	The Problems of Managing Software	3
2.3	Software Configuration Management	6
2.3.1	Basic Concepts and Terminology of SCM	6
2.3.2	SCM in Perspective	8
2.3.3	Software Configuration Identification	10
2.3.4	Software Configuration Control	11
2.3.5	Software Configuration Auditing	15
2.3.6	Software Configuration Status Accounting	15
2.4	Benefits of SCM	16
3.	REQUIREMENTS	17
3.1	The Software Environment	18
4.	EXPLORING THE STATE OF SCM	19
4.1	Evaluating Standards	19
4.2	Criteria for Evaluating Standards	19
5.	STANDARDS AND THEIR ASSESSMENT	22
5.1	British Standard 6488: 1984 Configuration Management of Computer-Based Systems	22
5.2	MIL-STD-483 (USAF) Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs	26
5.3	DOD-STD-2167, Defence System Software Development	30
5.4	ANSI/IEEE Std 828-1983, Software Configuration Management Plans	34
6.	AUTOMATED TOOLS FOR SUPPORTING SCM	37
6.1	Unix SCCS/MAKE	37
6.2	Unix RCS	38
6.3	DEC CMS/MMS	38
6.4	Oregon Software SourceTools	38
6.5	Softool CCC	38
7.	STANDARDS AND PRACTICES RECOMMENDATION FOR THE CANADIAN NUCLEAR POWER INDUSTRY	40

APPENDIX A	STANDARDS BODY ADDRESSES AND DATA	43
------------	---	----

APPENDIX B	ACTUAL USE OF SCM	44
------------	-----------------------------	----

B.1	Transport Canada Air Traffic Control Systems	44
B.2	Department of National Defence: DMCS-8	46
B.3	Bell Labs, Electronic Switching Systems	47
B.3.1	No. 4 ESS	47
B.3.2	TSPS No. 1	48
B.3.3	TSPS No. 1B	49
B.3.4	5ESS	50

APPENDIX C	BIBLIOGRAPHY	51
------------	------------------------	----

FIGURES

2.1	SCM in Perspective	5
2.2	Software Change Control - Example 1 (Operational Product)	13
2.3	Software Change Control - Example 2 (New Product)	14

1. INTRODUCTION

Computers now play an important role in the operation of nuclear powered electricity generation stations in Canada. They perform essential functions in reactor monitoring and control. Computers are also integrated into reactor special safety systems.

Computer software should be correct, efficient, reliable¹, and maintainable in all station computers. Station safety, reliability, and efficiency depends on this quality. Consequently, the AECS must apply some means of software quality assessment as a normal practice in the evaluation of licensee operations.

Emphasis on software quality is prevalent in all areas where software performs critical functions. This has resulted in the emergence of a more formal software quality assurance discipline. The practice of software configuration management (SCM) has grown in response to the requirement for software quality assurance. SCM is now considered as an indispensable tool for providing software quality assurance (SQA).

The increasing complexity of the software used in nuclear power stations demands that the SCM discipline be assessed with regard to benefits of its application. This report reviews the nature of SCM, its benefits, and its current status in terms of industry and military standards. The report further examines available automated software tools that support SCM and the experience of several Canadian organizations that are involved in the production and maintenance of critical software.

The recommendations of this report have been made without regard to the current or proposed activities of AECS licensees, and are not intended to imply that any deficiencies exist in these activities. Rather this report is intended to provide an independent review of SCM for real-time software in nuclear power stations.

¹ Software reliability is the probability that software will not cause the failure of a system for a specified time under specified conditions. The probability is a function of the inputs to and use of the system as well as a function of the existence of faults in the software. The inputs to the system determine whether existing faults, if any, are encountered.

2. MANAGING SOFTWARE

Any organization that is responsible for the production or maintenance of systems is faced with the need to make changes to correct recognized deficiencies or to accommodate new requirements. These changes can tend to degrade the overall integrity of the system. The problem stems from having to manage a large number of interrelated system components. A change in one component may require a corresponding change to be made in a related component. If both changes are made at a central maintenance facility, but only one of the updates is installed at the operational field sites, then the operational sites may become less reliable. In this way an attempt to improve safety may in fact reduce it. Such problems are addressed by the discipline of configuration management.

2.1 Configuration Management

Configuration management (CM) is the term applied to techniques used to coordinate the development and maintenance of a system. Bersoff [5] states that configuration management is:

"the discipline of identifying the configuration of a system at discrete points in time for the purpose of systematically controlling changes to the configuration and maintaining the integrity and traceability of the configuration throughout the system life cycle".

Configuration management is generally concerned with the consistent labeling, tracking, and change control of the components of a system. It has been traditionally applied to hardware systems or to the hardware elements of a hardware-software system. Existing configuration management guidance generally treats the hardware in detail, while software is handled as a single entity. This superficial treatment of the software subsystem suppresses its visibility during the evolution of the overall system. Lack of visibility has typically resulted in a corresponding lack of quality and integrity control.

Fundamental differences in the nature of hardware and software have led to failures in applying conventional CM practices to software systems. Software is represented by coded statements in a programming language and preserved on tapes or disks, but the actual statements themselves are abstractions, even in their operational state. Hardware, on the other hand, can be touched and seen. The concept of hardware maintenance cannot be directly applied to software, since software (the abstraction, not the

representation or the medium) does not wear out. Finally, software products seem to be much more individualized than hardware products, with fewer conventions and standards.

This situation has prompted the recent development of the discipline of software configuration management.

2.2 The Problems of Managing Software

The definition of software has been evolving as a result of attempts to define SCM. It has always been agreed that the coded statements of a program are software, but a consensus has developed lately that all associated documentation (specifications, design documents, user manuals) and related test programs and test data is also software.

According to this definition, software is coded information, that is easily changed - as suggested by the prefix "soft". The fact that software is easy to change has significant implications with respect to SCM. This susceptibility to change, this plasticity, is the primary reason why disciplining the software development and maintenance processes is essential.

Another view of software is revealed when examining the process by which ideas are transformed, during software development, into something that makes the computer hardware operate. Software exists in two basic forms:

1. A nonexecutable form, which can be further classified as: (a) documentation such as functional specifications, design specifications, flow charts, listings of source code, and (b) information embedded in machine readable carriers such as punched paper tapes, magnetic tapes, disk packs and other removable storage media that carry computer programs and supporting data.
2. An executable form, which consists of executable sequences of information directly accessible by, and unique to, a given suite of computing machinery (eg. sequences of compiled machine instructions in memory, or sets of data values in a disk file).

Software must be converted (ie. compiled, translated, copied) from the non-executable form to the executable form in order to have it control the operation of a computer. It has proven unsafe to have programmers directly change the executable form. The non-executable form may be distributed on many different

carriers, there may be many different versions of it available reflecting different levels of approved changes, and the steps needed to make each version executable may differ. It is generally very difficult to keep the non-executable representations of the software and the executable form all consistent. This is particularly difficult if some changes must be distributed quickly to all operational sites. The only method available for doing this is the use of clearly defined change control procedures that produce audit trails which permit checking by independent reviewers to ensure that all the required steps have been followed to maintain consistency in all forms of the software. SCM is the application of these disciplined methods.

Some of the practical everyday problems that arise in managing software are:

The Multiple Maintenance Problem

When, for example, there are two copies of the same software (eg. the same module present in two different systems), then both copies need to be maintained. When an error is found, someone needs to remember to fix it in both copies. Failing that, the error surfaces independently in each copy, and it must be diagnosed and repaired twice. Even when staff are instructed to fix both copies, eventually someone will forget. In general, multiple copies often diverge.

The Shared Data Problem

When several programmers are directly modifying a single copy of a program, then changes made by one programmer can interfere with the progress of others.

The Simultaneous Update Problem

This arises when several programmers are changing the same software that is held in a library. If there is uncontrolled access to the library, each programmer copies the library software module to his own work space, makes his changes to it, and copies the updated version back to the library. What can happen is that one programmer's updates will be lost (overwritten). An error that was fixed yesterday suddenly becomes unfixed today. More complex consequences can arise when a programmer makes a change that spans multiple modules.

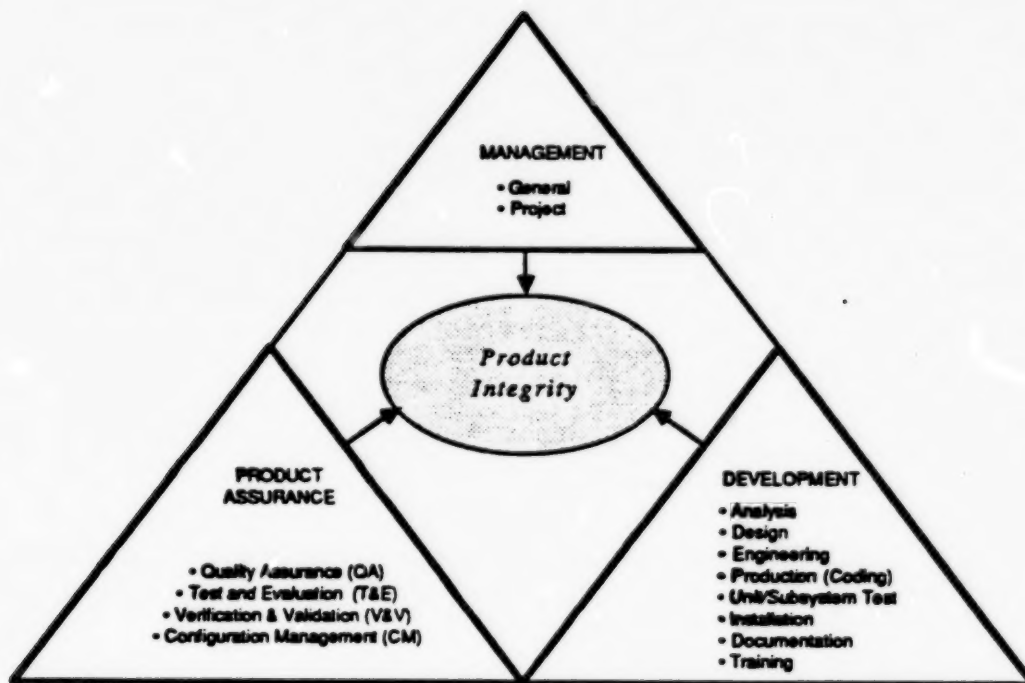


Figure 2.1
SCM in Perspective
(from Bersoff [5])

2.3 Software Configuration Management

The recently emerged discipline of SCM has its roots in the traditional configuration management practiced in the hardware domain. As a result SCM is viewed as consisting of the same four main functions:

- configuration identification
- configuration control
- configuration auditing, and
- configuration status accounting.

These functions will be further explained below.

Configuration management when applied to software has a different emphasis than when applied to hardware. SCM is part of the software engineering process, it comes into play during design and is carried out during all stages of maintenance. This is generally accomplished using techniques of multiple levels of baselines and internal releases (or promotions) to a larger degree than is typically applied by hardware configuration management. SCM directly manages the computer program's representations whereas hardware CM focuses mainly on the management of documentation.

Before looking into the details of SCM's procedures it is vital to understand its relationship to the other aspects of software development and maintenance.

2.3.1 Basic Concepts and Terminology of SCM

This section defines and explains some of the useful concepts in SCM.

Configuration item (CI)

A collection of hardware or software elements designated for purpose of configuration management. (Further described in 2.3.2 below)

Software configuration

The complete set of all the source code, object code, and related documentation of a particular version of a software configuration item.

Baseline

As used in most SCM standards, the term baseline refers to a specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

A baseline will contain a group of related documents or software components that are part of the same configuration item that have all been captured, labeled and somehow denoted or recorded as parts of the same baseline. A baseline generally continues to exist for the life of the configuration item. Applying approved changes to a baseline produces a new version or revision of the same baseline (not a new baseline).

There are three formal baselines that a configuration item can have. The functional baseline contains all documents that define the agreement between the developers and the users that define all functions at a system level. The allocated baseline is the agreement between the developers and users that identifies all the software requirements and includes design constraints and user-required standards. The product baseline is the agreement between the developer and user that defines the exact version of the software product that is delivered.

During development there can be a number of informal baselines called developmental baselines.

Baselines are an effective mechanism for synchronizing people working on the same project together at the same time. SCM focuses its activity around the construction and maintenance of baselines.

Version

This term is most often loosely applied to the outcome of applying a change to a software component or system. But when applied more strictly for purposes of identifying software, the term indicates a software configuration item with a defined set of functional capabilities. As functional capabilities are added, modified, or deleted the CI is given a different version identifier.

Revision

This term is applied to a new instance of a program that has been changed only to correct errors in the design or code without affecting its documented functional capability.

Variant

One of a family of related programs that have subtle differences. For example, different variants of a software item may be used to adapt to differences in peripheral hardware in different computers.

Release

This term is applied to the act of handing over a controlled entity (eg. a CI, source code, a document) from one organization or team to another during the development process. A release also indicates that there is a transition in level of authority needed to approve changes to the entity. The term is most often used when an entity moves into a formal baseline (eg. a software component released to users in the product baseline).

Promotion

When internal or developmental baselines are created as a part of the engineering process and entities are moved or released to another internal activity for additional work, integration or testing, the term promotion is used to distinguish this type of release from the more formal releases to users.

2.3.2 SCM in Perspective

Software configuration management plays its role in the wider context of software development or maintenance. The process of producing or modifying software is dominated by the goal of creating software with "product integrity". Software is said to have product integrity if it fulfills the user's functional needs, meets other specified criteria (eg. performance), is produced within expectations of cost and schedule, and can be easily and completely traced through its life cycle. This last characteristic provides the user with visibility of the product from its point of conception, during its development, and throughout its subsequent evolutionary modifications.

Organizations that undertake the development of software are usually arranged under three basic functional disciplines:

- project management,
- development, and
- product assurance.

Figure 2.1 depicts the relationship of these disciplines for the attainment of product integrity. It is important to note that

configuration management works in concert with the other product assurance activities of quality assurance (QA), test and evaluation (T&E), and verification and validation (V&V) to support the development and management disciplines. T&E is the activity carried out independently of the development group to assess, by execution of a set of test plans and procedures, whether a product fulfills objectives. V&V addresses how well software fulfills functional and performance requirements and how well the requirements are stated and interpreted. (See references [4] and [5] for a more detailed discussion of organizing for product integrity.)

Because of the apparent separation of SCM from development there may be a tendency to view product assurance as having a negative impact on development productivity. This is not always true. Some SCM procedures, especially if they are supported by automated tools, can provide a significant enhancement of productivity of a development team by enforcing controlled coordination. SCM should be viewed as service function supporting both technical and managerial activities making everybody's job easier. Effectiveness of the SCM processes increases in proportion to the degree that the discipline is a part of the normal day-to-day activities of everyone involved in the development and maintenance effort.

To illustrate SCM's role in an organization consider, for example, the activities surrounding the testing of a software module. The development team performs the test, then submits the test program, data, results and report to the SCM team. The test products thus become part of the overall software product and are thus managed by SCM (eg., identified, controlled, traced to other entities). The SCM team may verify that the test was solicited by a change order (ie. was authorized) then request the SQA team to sanction the test before storing the test products in the project database. The SCM team will finally perform the necessary status accounting to record that the test is complete and to trace it to change orders and other documents.

Configuration management also exhibits a dual relationship with quality assurance. It provides support for the QA function in terms of information reports about the system configuration and the state of all changes that have been or are being applied to the configuration. On the other hand, quality assurance must be applied to the configuration management process itself. This can be done through routine audits of the process to assess its effectiveness and conformance to a pre-specified plan.

All SCM standards demand that a formal SCM plan be written and approved at the onset of each new software project. This plan embodies all considerations of management, procedures, and

10 August 1987

resources that will be used for the project's SCM.

The SCM plan, or the larger system definition in which the plan operates, must define the software entity that is to be managed. For example, military systems are divided into hardware and software configuration items (CIs) over which separate and independent configuration management is imposed. Software CIs are eventually further divided into a hierarchy of components and units. The size of a software CI can vary greatly. Many factors need to be considered in the selection of a software CI. (See MIL-STD-483, for selection guidance.) For example, a software CI could cover all the software in all the computers and programmable digital comparators of an entire nuclear power plant. Another choice may be just one type of software (eg. application, support, diagnostic) in one particular computer or class of computers. The weighing of CI selection criteria will generally be quite different from one organization to another.

The form and style of SCM chosen by an organization is strongly determined by its software life cycle concept. For example, in the military environment, software is initially developed by a contracted supplier through a series of well defined phases which terminate with deployment of operational software and associated documentation. SCM operates throughout the whole process and imposes additional tasks at the start and end of each phase. When subsequent changes to the software are sought, the whole development cycle is restarted. This iteration of cycles continues for the life of the software.

In other communities, software may be viewed as undergoing continuous evolution throughout its life, punctuated by release, delivery or installation of new software versions. Here SCM identifies, controls, and tracks all changes that form the new versions.

2.3.3 Software Configuration Identification

This function is responsible for defining and labelling the software product being managed. In the first stages of development, the product may only consist of a requirements specification document. As the system evolves, components are added and changed. Periodically the configuration will exhibit the properties of usefulness that were intended by the development team. At these points, the configuration identification procedures are invoked to document and record the precise configuration of components in what is often called a baseline.

10 August 1987

Changes that are applied to a baselined configuration must also be recorded so that at any moment it is possible to answer any query about the current configuration. The function thus brings essential visibility to otherwise intangible software.

2.3.4 Software Configuration Control

The role of software configuration control is to provide the administrative mechanism for precipitating, preparing, evaluating, and approving or disapproving all proposals for change to the software configuration. The procedural arrangements to accomplish these tasks can vary widely and must be tailored to the workings of the organization that will be using them.

The communication of change information between the configuration control tasks is mediated by special forms and documents. One of the most common of these is the engineering change proposal (ECP) which documents the need for a change and all anticipated aspects of the impact of the change so that an authoritative body (eg. configuration control board) can make an informed approval or disapproval of the change.

Computer automation can be used to great advantage in this function for handling the clerical duties and especially for providing a software library with built-in protection, security and archiving mechanisms.

Two possible plans for the flow of information in the configuration control process are depicted in Figures 2.2 and 2.3. The first example might be adopted by an organization that is maintaining a software product that is in active use in an operational system. Unsatisfactory condition reports (UCRs) and change proposals (CPs) generally originate from the users of the operational system. The plan allows for resulting changes which are classified to be only minor corrective changes or adjustments to be incorporated into the operational software without configuration control board (CCB) approval. Management has required that other changes that have greater economic or operational impact be routed to the CCB for formal approval.

The second example, in Figure 2.3, might be applied by an organization during the development of a new software product or in performing a major upgrade of an existing product. In this situation most software change requests originate from the development team when they recognize deficiencies during design, coding or testing. Here the CCB would be manned by technically knowledgeable people who would place more emphasis on the system

and software implications of changes rather than on higher level management concerns. In addition to approving or disapproving a change request, the CCB can also table a change request in order to defer it for further analysis or allocation of resources.

10 August 1987

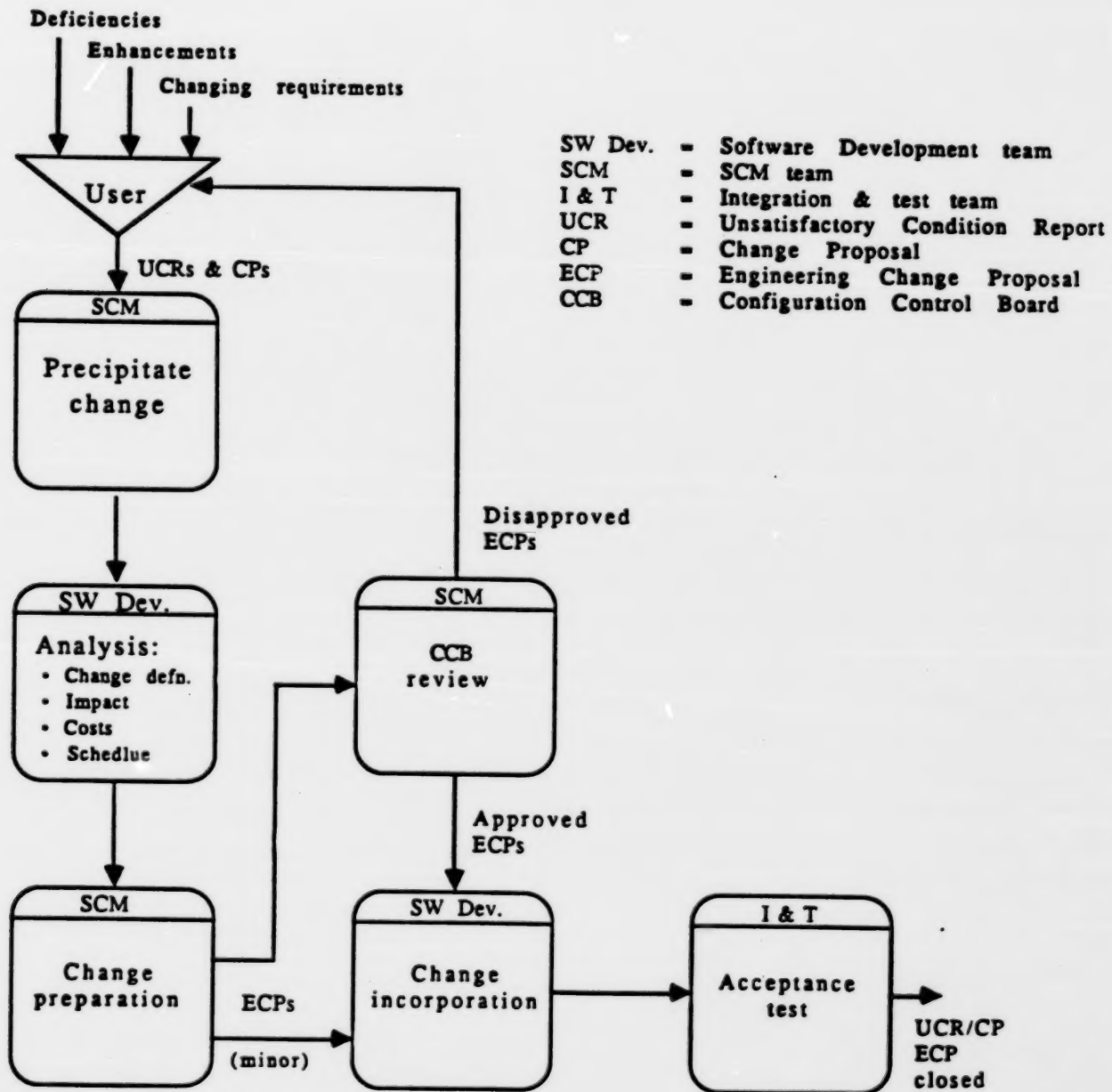


Figure 2.2
 Software Change Control - Example 1 (Operational Product)

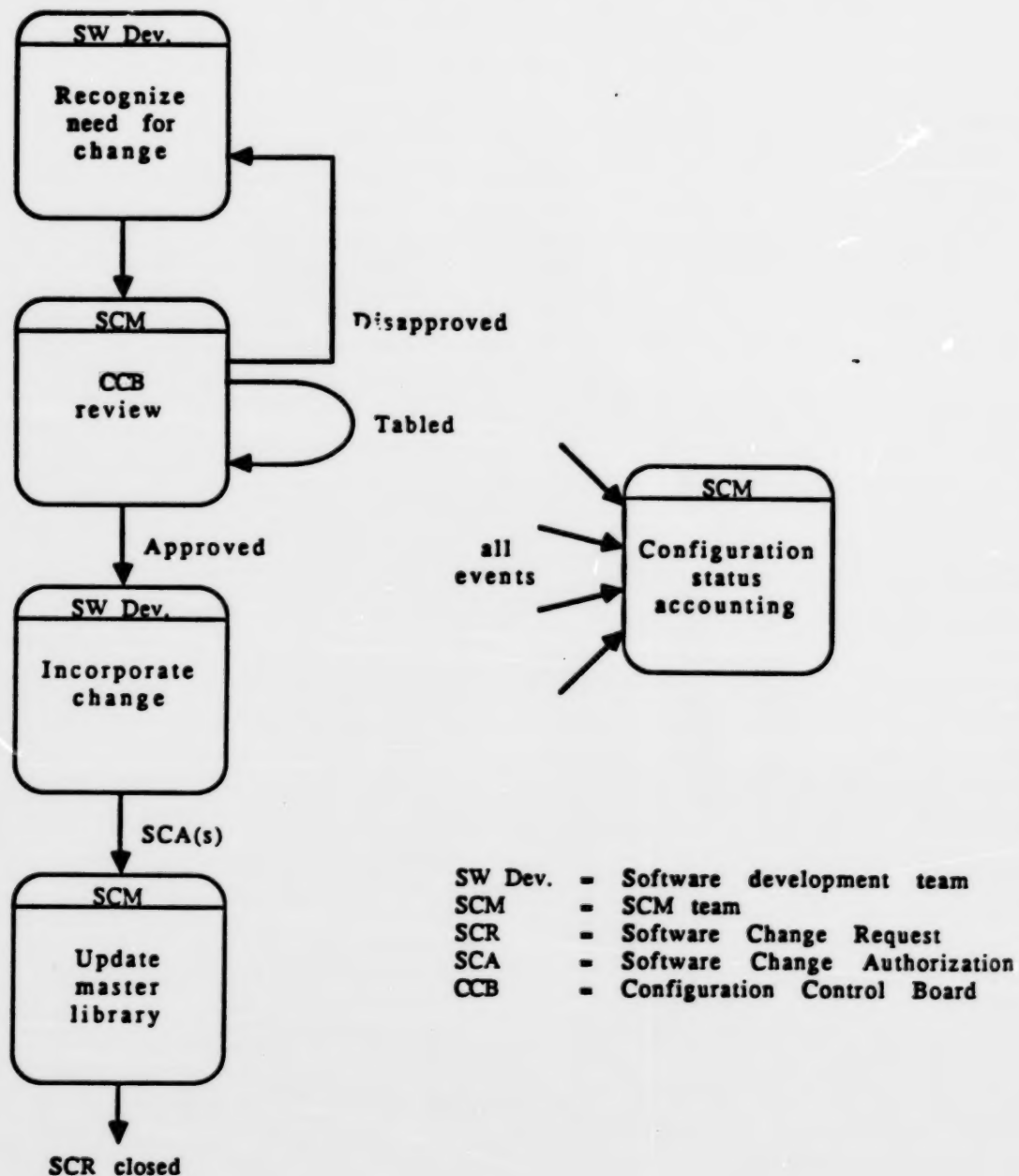


Figure 2.3
 Software Change Control - Example 2 (New Product)

2.3.5 Software Configuration Auditing

The software development practices adopted today by most industry organizations involved in the production of critical software call for a series of reviews and audits as part of the development process. Reviews are used to mark the end of each major development phase and give important visibility to the customer, user and management.

Audits are one means by which an organization can ensure that the developers have done all their work in a way that will satisfy any external obligations. Audits vary in formality and rigor, depending on the legal liability of external obligations, ranging from customer contracts to in-house users. They are a final check on the completeness of a computer program product.

Generally there should be a physical configuration audit (PCA) and a functional configuration audit (FCA) prior to the release of the product or updated version of the product. The PCA determines that all items identified as part of the configuration are present in their required versions. The FCA involves confirmation that each item has been inspected or tested to determine that it satisfies the functions defined in the specification to which the product was developed.

The SCM team generally does not initiate or direct reviews and audits but instead provides necessary support for these activities. Quite often the mechanisms used by SCM to process changes are used to organize and process items in a review conducted by other teams such as SQA.

2.3.6 Software Configuration Status Accounting

This function involves the collection, storage, processing, and reporting of information about the state of the software configuration and all changes whether proposed, outstanding, or completed. The database of this information is used to support all the other facets of SCM and any special status accounting requirements specified by the ultimate software user or other agencies.

2.4 Benefits of SCM

The practice of SCM results in:

1. improved allocation of software development resources through:
 - reduced duplication of effort
 - maximum reuse of common software components
 - improved accounting of software change costs
 - priority given to most important changes
2. improved reliability of the operational software through:
 - procedures to manage several versions of the software
 - complete recording of change histories
 - wider propagation of changes to all relevant sites
 - complete visibility of all outstanding known software deficiencies
 - single point of responsibility for software configuration
 - improved visibility of the status of changes being made at all sites.

The subject of software reliability touches on the important issue of safety whenever the software is involved in critical processes where a failure could result in death, injury, loss of property, or environmental harm. Some of the safety problems that are addressed by an effective SCM plan are:

- wrong software installed in an operational computer
- an erroneous combination of software components installed in an operational computer
- insufficient dissemination of information on known software deficiencies
- safety improvements developed at one site are not promulgated to other sites
- insufficient review of proposed changes leading to installation of changes that reduce safety
- inadequate visibility of a software configuration to effectively support the quality assurance function
- poor documentation of dependencies between components of a configuration (a condition which could allow a software change to be made that causes a detrimental effect in other areas of the software)

3. REQUIREMENTS

This section lists the requirements of the Canadian nuclear power industry that must be addressed when considering software configuration management for the industry.

1. The credibility of the nuclear regulatory process depends not only upon its technical correctness and practicability, but also upon acceptance by the public-at-large of its perceived effectiveness and efficiency. Accordingly, the selected SCM practices should constitute a blend of industry standard and state-of-the-art SCM procedures that are acceptable to the technical community at large and which convey a conscientious and comprehensive approach to safety-related aspects of the nuclear power industry.
2. The selected SCM elements should take recognition of the variable size, criticality, and complexities of the range of software to be managed. Via customization, the SCM procedures should be applicable and amenable to both large and small, long and short-term projects. In short, the program should be applicable to a full range of software projects and all life cycle phases.
3. Software engineering is a dynamic discipline, and considerable development in SCM has occurred in recent years. Accordingly the SCM elements selected should be, as far as possible, flexible and accommodating to future improvements.
4. The majority of AECS licensees currently use configuration management for hardware. The selected SCM elements should be capable of implementation within the general configuration management approach already in use, while also catering for unique requirements of the software life cycle.
5. An important safety aspect of the nuclear industry is the complete recording, reporting, and analysis of untoward events that occur during commissioning and operation. SCM procedures should ensure full visibility into such incidents where software is involved.

3.1 The Software Environment

This section briefly summarizes the environment in which nuclear generating station computer software is developed, operates, and is maintained insofar as it affects the configuration management treatment of the software.

The main points about the software environment are:

- Generating stations exist at geographically separated sites and are owned and operated by three different power utility companies.
- A station may have several reactor units and each unit has many operational computers.
- The models of computers used at one reactor unit may be quite different from those used at other reactor units of the same station and at other stations.
- There are two main classes of computers used at a station: control and monitoring computers, and microprocessors used in the special safety systems. Software for the latter class is stored in read only memory (ROM) devices.
- Software for the safety system computers is currently being maintained by Atomic Energy of Canada Ltd. (AECL).
- Most of the software for control and monitoring computers is currently being maintained at the individual generating stations.
- The software source code is maintained on a variety of mini-computers and personal computers at the stations.
- At present there is no formal exchange between stations of software, deficiency reports, or change notices.

10 August 1987

4. EXPLORING THE STATE OF SCM

Having acknowledged that disciplined software configuration management is beneficial for the Canadian nuclear power industry, our attention should now turn to exploring the current state of SCM practice within the software industry in general. We start by surveying the most immediately available resources: industry and military standards.

4.1 Evaluating Standards

One of the first steps in exploring the current state of SCM is to examine the available standards on the subject. Adopting recognized standards offers many advantages.

- Standards represent a consensus of good engineering practice which may have a legal bearing in liability considerations.
- Standards are periodically reviewed and updated as technology advances and as practical experience dictates.
- Standards facilitate better communication among the community of users and experts.
- Standards ease the sharing of procedures, tools, and other resources with other users.
- Personnel education, training and experience will be more prevalent in the direction of standards.

4.2 Criteria for Evaluating Standards

The following is a list of the criteria that are applied in this report for the evaluation of software configuration management standards and practices.

- a) Nature of the developing body
The organization that has developed the standard should be considered in the light of its goals and motives.
- b) Assumptions
How closely the implied or explicit assumptions of the standard match the conditions of the Canadian nuclear power industry.

- c) Usability
Ease with which the stated SCM practices can be incorporated into the operation of the nuclear power industry (personnel involvement, management organization, required training, etc.).
- d) Customizability
Ability to modify or extend the standard to meet the special needs of the Canadian nuclear energy industry.
- e) Distributed management
This relates to the standard's applicability to an environment where the same software configuration items may be maintained by several diverse organizations (eg. information dissemination, responsibility sharing, etc).
- f) Development vs. maintenance
It is generally desirable that the same practices be usable both during the initial development of software and its subsequent post-delivery maintenance phase.
- g) Multi-variant treatment
The characteristics of the standard in regard to the reality of having to simultaneously maintain several variants of the same software. This happens because software components may be at different stages of enhancement or may be tailored for different sites.
- h) Response time
The effect that a SCM plan based on the standard may have on the time in which identified software deficiencies can be responded to.
- i) Temporary/emergency changes
Provisions in the standard that make allowance for expeditious treatment of emergency software deficiency reports.
- j) Documentation
The features of a standard that promote the recording of costs and resource usage against maintenance activities, the generation of documents and forms that are useful for status accounting, tracing, etc.
- k) Terminology
Conformance to industry and military conventions of terminology used in the SCM field.

- l) **Comprehensiveness**
Does the standard address all aspects of SCM as it is currently understood?
- m) **Level of detail**
Does the standard treat each topic in sufficient detail to leave little room for misinterpretation or confusion. A high level of detail can provide the material for formulating a good SCM plan providing that it does not hinder customizability.
- n) **Assessability**
Features of the standard that allow for a continuous or periodic assessment of the effectiveness of SCM plans.
- o) **Support for quality assurance**
Will the QA organization be able to make good use of the status accounting and auditing features of the plan to evaluate the quality of operational software?
- p) **Automation**
Ability to support the standard with automated tools.
- q) **Cost**
The cost (expenditure and time) of developing a plan, establishing it, and carrying out its stipulated procedures from day to day.

10 August 1987

5. STANDARDS AND THEIR ASSESSMENT

This section analyses four SCM standards using the evaluation criteria listed in the previous section.

5.1 British Standard 6488: 1984 Configuration Management of Computer-Based Systems

This British standard code of practice provides a brief but good general set of criteria on which to base a software configuration management plan. However, it lacks considerable detail, even as a check list, for formulating a practical plan.

Some of its distinguishing characteristics are:

- In addition to hardware and software the standard considers services and supplies to also be part of a system.
- Documentation is not viewed as being part of the system, but is considered to be a product of configuration identification in that it describes or is a textual representation of the configuration.
- Paragraph 8 (g) (3) of the standard introduces the concept of a change that is approved but is not to be implemented until some future specified condition is reached. This notion does not appear in other standards.
- The treatment of deficiency reports are not addressed. The change process only starts with change proposals.

Evaluation:

Nature of the developing body

This standard was produced by a committee of representatives from almost every area of the British community of computer user organizations, including the Electricity Supply Industry in England and Wales.

Assumptions

The assumption is made that the systems will be documented to a certain standard level and that the documentation will be maintained. Testing is also expected to be carried out in compliance with a specific standard. It is also assumed that the system will be subjected to performance monitoring.

10 August 1987

The standard makes no special assumptions about the system life cycle concept. For example, it does not require that changes to the system be treated as new development cycles (as in the U.S. military standards).

Usability

BS 6488 specifies a fairly generic and flexible style of SCM that should fit well into any organization. No specific procedures or personnel are identified, only that these issues should be defined in the SCM plan.

Customizability

The standard provides criteria on which to base an SCM plan. It provides a short list of features that should be included in an SCM plan. On the whole, SCM plans for the Canadian nuclear power industry could be developed that satisfy the standard's criteria, but it would be beneficial to consult other sources for more detailed advice.

Distributed management

BS 6488 does address the notion of geographically dispersed systems and development staff. Although it requires that the "Master Configuration Index" be centrally maintained, it provides for copies of it to be distributed to distant locations.

Development vs. maintenance

The standard attempts to define SCM over the complete life span of the system commencing with the system requirement specification and continuing until the system is discontinued and its documentation destroyed. The in-service life of the system is viewed as punctuated by "reference points" at which the configuration is baselined. It makes no requirement that change activity be treated as new development, although it can be incorporated by organizations that do manage changes in this way.

Multi-variant treatment

The standard requires that the master configuration index document the relationships between multiple application systems and multiple installations when these exist. However it fails to give guidelines for dealing with multiple variations of the system, either existing in use in the field, or as multiple configurations under development or enhancement.

Response time

The procedures for dealing with software deficiencies are not defined in sufficient detail to allow comment on how they might affect the time to respond to reported deficiencies.

Temporary/emergency changes

This area is not addressed.

Documentation

BS 6488, for the most part, demands the normal SCM documentation items in a general way. However it is specific in its requirement for a master configuration index document which must be produced for each defined configuration baseline.

This standard is also unique in that it considers what is usually viewed as system documentation to be a product of configuration identification. In other words, system documentation is not itself under configuration control but is instead used as an extension of the master configuration index to describe the system that is under configuration control.

The standard also requires that the organization have a documentation library control system.

Terminology

BS 6488 introduces a few unusual terms:

- The "Master Configuration Index" can be viewed as the equivalent of the "Version Description Document" specified in other standards.
- The term "Reference Point" is used to define the times in a system's development when a baseline is defined.

Comprehensiveness

The major areas of SCM are covered by this standard, but only at a superficial level. It provides no guidance for the processing of system deficiency reports -- the generation of change proposals is outside the scope of the standard.

Level of detail

The standard offers very little detail for establishing an SCM plan.

Assessability

The standard offers a brief recommendation for auditing of the configuration management functions. No special provisions are made in the rest of the standard to facilitate this task.

Support for quality assurance

The standard advocates the production of the usual documentation items that can serve as resources for the quality assurance function.

Automation

No particular requirements of BS 6488 interfere with the implementation of its various procedures on automated systems. The standard recognizes and makes reference to the desirability of using machine facilities.

Cost

The standard does not offer sufficient detail to assess the cost impact of implementing an SCM plan based on its requirements.

5.2 MIL-STD-483 (USAF) Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs

This U.S. military standard sets forth configuration management practices that can be stipulated in procurement contracts for hardware and software. It is supported by various appendices and other military standards that together form a richly detailed set of requirements for configuration management, quality assurance, and specifications.

The standard has a wide scope that goes beyond the usual domain of configuration management and deals with many other areas such as system specification, design, development, testing, review activities, contract management, and quality assurance.

Some of the key standards that MIL-STD-483 references are:

- | | |
|-------------|--|
| DOD-STD-480 | Configuration Control - Engineering Changes, Deviations and Waivers |
| MIL-STD-481 | Configuration Control - Engineering Changes, Deviations and Waivers (Short Form) |
| MIL-STD-482 | Configuration Status Accounting Data Elements and Related Features |

MIL-STD-483 together with its referenced standards have been evaluated as a family of standards.

Evaluation:

Nature of the developing body

This standard was developed by the U.S. Air Force Systems Command and is available for use by all departments and agencies of the Department of Defence. It is written from the point of view of a procuring agency that must specify a product in such a way as to have control over its design, development and delivery.

Assumptions

MIL-STD-483 is assumed to be applied by a procuring agency by reference in a contract statement of work. Despite the effect that this assumption has on the wording of the standard, it can still be applied by any organization as part of its internal operation.

The standard relies on U.S. military system development

10 August 1987

concepts such as baseline management, system segments, interface control, specification practices, and identification numbering.

Usability

MIL-STD-483 and its related standards would be difficult to use in their present form. The family of standards is very large and complex. However, it can serve as a very useful resource for the preparation of a detailed SCM plan.

Customizability

As it is written, the standard is intended to be customized for a particular procurement activity by explicitly identifying within the procurement contract paragraphs that do or do not apply.

For practical use by the Canadian nuclear power industry, the standard would have to be extensively rewritten to take into account the special operating environment.

Distributed management

The standard addresses this issue only as it relates to the relationships between a procuring agency, a prime contractor, and sub-contractors. Responsibilities of these parties are fully defined and detailed procedures for information exchange are laid out.

Development vs. maintenance

The major emphasis of the MIL-STD-483 is on the initial development of a configuration item and no particular attention is paid to continued maintenance of software. This is because, in the wider life cycle view, changes to a completed system are treated as new procurement activities.

Multi-variant treatment

No provisions are made for managing multiple variants of the same software.

Response time

The standard (through DOD-STD-480) stipulates in considerable detail the process by which engineering change proposals (ECPs) must be prepared, submitted, reviewed and approved. The high degree of documentation that must accompany a formal ECP may be viewed as contributing to delays in approval response, however such information is generally essential to making an informed approval.

Temporary/emergency changes

Special provisions are made in DOD-STD-480 for the handling of emergency and urgent engineering change proposals.

Documentation

The standard requires the production of many different types of documentation to support the change process. The most important of these are the Engineering Change Proposal (ECP) and the Specification Change Notice (SCN).

Terminology

The standard conforms to the commonly used industry and military conventions in North America for SCM terminology.

Comprehensiveness

MIL-STD-483 and its related standards fully cover all aspects of SCM except in areas of extended maintenance to operational software, and the multi-variant problem.

Level of detail

Most areas of the MIL-STD-483 family of standards are presented at a very high level of detail. The high volume and complexity of material requires that the reader spend substantial time to become familiar enough with the standards before making effective use of them.

Assessability

The standard requires that the SCM plan describe how configuration audits will be conducted. Both functional and physical configuration audits are described in the standard but these descriptions are highly biased toward hardware procurement and are not too helpful in an environment where software is under long term maintenance.

Support for quality assurance

The wide scope of the standard covers areas of quality assurance in software development that in some circles would not be considered as part of SCM. For example, the requirements for the Computer Program Configuration Item Specification call for a special section (Section 4) on quality assurance provisions. This section details how the configuration item's requirements defined in Section 3 shall be formally verified. This can include a verification cross-reference matrix that can be directly used for check-off by QA staff. Such a matrix defines, for each requirement of Section 3, the type of verification and test procedure that shall be applied, and the associated verification requirement from Section 4.

Within the usual SCM domain, the standard requires that

configuration audits provide a check to ensure that acceptance testing will be adequate for the quality assurance activities.

Automation

For the most part, there is no special consideration for the use of automation to reduce the workload of the SCM procedures. However, in the area of configuration status accounting, an automated system could make use of the standard formats and designations for the fields of data that appear in status accounting records and reports that are defined in MIL-STD-482.

Cost

Full adherence to MIL-STD-483 and its associated standards would undoubtedly impose a significant cost burden on any organization which is not otherwise required to conform to U.S. military standards.

10 August 1987

5.3 DOD-STD-2167, (June-1985)
Defence System Software Development

DOD-STD-2167 is a recently developed U.S. military standard that addresses all aspects of software development for mission critical computers. These are computers embedded in certain specified military systems. The new draft revision of the standard broadens the scope of applicability to include all types of computer systems.

Requirements for software configuration management are confined to Section 5.7 of the document but many other aspects of software development that appear throughout the standard are related to the SCM task.

This standard represents a considerable improvement over previous standards because it attempts to state only basic requirements without going into how to achieve the requirements. However, for purposes of guidance and further detail, it makes reference to the MIL-STD-483 family of standards. DOD-STD-2167 also offers a better treatment of configuration management for software than the older standards which were mainly oriented to hardware.

Another important feature of the standard is the many data item descriptions (DIDs) that are referenced by the main document. Each of these give specific details of the required format and content of various documents that must be created and maintained as part of the software development. Of particular interest for SCM are: DI-MCCR-80009 which specifies the Software Configuration Management Plan, and DI-MCCR-80013 which specifies the Version Description Document.

As with other military standards, DOD-STD-2167 is intended to be used for establishing software development requirements as part of a government contract for acquisition or support of software or a system containing software. It is intended also that the standard will be tailored for each contract by the deletion of non-applicable requirements.

A new revision to the standard is now in preparation. The latest draft, DOD-STD-2167A, 1 April 1987, makes considerable changes over the original version. Overall, it further removes "how to" information by just stating basic requirements. In the area of SCM, the new draft version removes the tasks of reporting and correcting software problems from the domain of SCM and places them under software development management.

10 August 1987

Evaluation:

Nature of the developing body

This standard was developed by the U.S. Department of Defence to specify a uniform software development process that will be cost-effective over the entire life cycle of acquired software. The standard was originally addressed at software for mission-critical computer systems but later revisions of the standard have dropped this emphasis.

Assumptions

DOD-STD-2167 is assumed to be applied by a procuring agency by reference in a contract statement of work. Despite the effect that this assumption has on the wording of the standard, it can still be applied by any organization as part of its internal operation.

The standard relies on many of the U.S. military system development concepts that MIL-STD-483 does, but to a lesser extent.

Usability

DOD-STD-2167 is a clearer and more understandable standard than its predecessors. It provides enough detail to form the basis for an SCM plan for the nuclear power industry.

Customizability

DOD-STD-2167 is designed to be specifically customized or tailored for each application. It even contains an appendix that gives clear guidelines for tailoring the standard. The requirements for the SCM plan allow it to conform to an existing organization, personnel, procedures, and resource availability.

Distributed management

DOD-STD-2167 specifies two levels of configuration management, one (performed by the government) which controls the functional baseline (eg., specifications of software requirements) and the product baseline (eg., operational software), whereas the other level is the responsibility of a contractor who maintains control over his own internal developmental configuration during a development or modification phase. Aside from this concept, there is no special treatment of distributed management.

Development vs. maintenance

Appendix B of the standard describes the system life cycle and the framework in which software development is conducted under the provisions of the standard. The assumption is that

10 August 1987

changes to an installed system will be accomplished through the same software development cycle that is used for the production of new software. In practice, many of the steps of most change cycles would be considerably less arduous than for full scale development.

Multi-variant treatment

No provisions are made for managing multiple versions of the same software.

Response time

The standard references MIL-STD-480 with regard to the processing of proposed changes which in turn imposes specific requirements for high priority changes.

Temporary/emergency changes

The standard makes reference to MIL-STD-480 for classification and processing of emergency and urgent proposed changes.

Documentation

DOD-STD-2167 requires the same type of SCM documentation as MIL-STD-483. The required forms and documents are:

- Problem/change report
- Engineering Change Proposal (ECP)
- Specification Change Notice (SCN)
- Version Description Document (VDD)
- Configuration status accounting reports

Terminology

The standard conforms to the commonly used industry and military conventions in North America for SCM terminology.

Comprehensiveness

DOD-STD-2167 and its related DIDs and standards fully cover all aspects of SCM except for the multi-variant problem.

Level of detail

DOD-STD-2167 provides a high level of detail through the referenced DIDs and standards. It does this without overwhelming the reader with detail in the main document. Also, the reader does not have to be entirely familiar with the family of MIL-STD-483 standards since DOD-STD-2167 references these only for expansion on certain narrow subjects.

Assessability

The standard requires that the SCM plan describe how configuration audits will be conducted and how the configuration status accounting reports and records assist in these audits.

Support for quality assurance

The standard supports QA by requiring the establishment of both formal and informal baselines against which QA monitoring is applied. It also mandates a corrective action system to handle QA detected deficiencies. The standard also requires the production of documentation items such as the SCM plan and version description documents that serve as resources for the quality assurance function.

Automation

DOD-STD-2167 encourages the use of automated tools in performing configuration management but specific requirements neither help nor hinder their use.

Cost

The cost of implementing SCM conformant to this standard will vary greatly based on the degree to which new procedures are introduced to the software development or maintenance organization. It should be kept in mind that the imposition of some SCM procedures can actually reduce overall costs by improving team productivity.

**5.4 ANSI/IEEE Std 828-1983,
Software Configuration Management Plans**

This standard, developed by representatives of industry, provides a well rounded approach to SCM that is simpler and more self-contained than the military standards reviewed above.

The standard is supplemented by two additional standards that are applicable to the nuclear power industry:

IEEE Std 1033-1985, Recommended Practice for application of IEEE Std 828 to Nuclear Power Generating Stations

ANSI/IEEE-ANS-7-4.3.2-1982, Application Criteria for Programmable Digital Computer Systems in Safety Systems of Nuclear Power Generating Stations

The first of these, IEEE Std 1033, simply contains the recommendation that there be a coordinated use of IEEE Std 828 and ANSI/IEEE/ANS-7432. It also recommends that IEEE Std 828 be fully applied to at least the software used in safety systems.

A special working group in the IEEE Computer Society has recently written a guide to software configuration management (ref [10]) that gives very practical, detailed assistance in developing and implementing SCM plans that are compatible with IEEE Std 828. The guide contains an encouraging wealth of helpful suggestions and detailed examples.

Evaluation:

Nature of the developing body

This standard was developed by a working group of the Software Engineering Subcommittee of the Technical Committee on Software Engineering of the IEEE Computer Society. The members, who serve voluntarily, are from various industry and standards organizations. The standard represents a consensus of the broad expertise of these members.

Assumptions

The standard assumes only that an organization or person is responsible for producing an SCM plan in preparation for a new software project. No assumptions are made about the context in which SCM will be performed or on the areas of the software life cycle that will be addressed.

10 August 1987

Usability

IEEE Std 828 imposes no requirements that would cause difficulty for any organization. All of the items that it recommends for inclusion into an SCM plan should be considered essential for the production and maintenance of critical software.

Customizability

The standard is essentially a valuable check list of items that should be considered for an SCM plan. The only firm requirement it imposes is the structure of the plan document in terms of section numbers and titles. The standard is also written to provide assistance to the SCM plan developer and offers many examples to enhance clarity and promote understanding.

Distributed management

Although this issue is not addressed specifically, the standard allows for an organization (or related organizations) to use multiple configuration control boards or teams to manage a software product. The SCM guide (ref [10], section 3.3.2.4) expands on this by discussing issues involved with the interfacing of configuration control boards.

Development vs. maintenance

The standard is entirely flexible on which phases of the software life cycle the SCM plan is directed to.

Multi-variant treatment

No support is offered in this area.

Response time

The procedures for dealing with software deficiencies must be defined by the developer of the SCM plan. The standard has no requirements or recommendations for these procedures or the response time they would incur.

Temporary/emergency changes

This area is not addressed.

Documentation

IEEE Std 828 contains implied requirements for the usual configuration status reports and other forms of SCM documentation.

Terminology

The standard conforms to the commonly used industry conventions for SCM terminology while avoiding much of the terminology used in the military standards.

10 August 1987

Comprehensiveness

IEEE Std 828 fully covers all aspects of SCM except in areas of the multi-variant problem and temporary/emergency changes.

Level of detail

The standard covers SCM issues to a considerable detail although in a very terse or checklist manner.

Assessability

The standard recommends that auditing of SCM activities be specified in the SCM plan.

Support for quality assurance

The standard supports QA by requiring the establishment of both formal and informal baselines against which QA monitoring is applied. The standard also requires the production of documentation items such as the SCM plan and configuration status reports that serve as resources for the quality assurance function. It also suggests that a quality assurance check be defined for software items before they are accepted into configuration management.

Automation

IEEE Std 828 goes further than the other standards in promoting the use of automated tools. Some of its requirements that would be perhaps best addressed with automation are in the areas of the software library system, the collection, storing, processing, and reporting of configuration status information, and supporting dynamic inquiries on configuration status.

Cost

The cost of implementing SCM conformant to this standard will vary greatly based on the degree to which new procedures are introduced to the software development or maintenance organization. It should be kept in mind that the imposition of some SCM procedures can actually reduce overall costs by improving team productivity.

6. AUTOMATED TOOLS FOR SUPPORTING SCM

This section describes a few of the commercially available SCM tools. It is intended to show a representative range of tool capabilities, and is not intended as an exhaustive survey of tools. See reference [3] for a description of other similar tools. The IEEE guide to SCM (ref [10]) defines four levels of automation that SCM tools can provide to the programming environment on a project.

6.1 Unix SCCS/MAKE

SCCS (Source Code Control System) and MAKE are software configuration management utility programs that are distributed with the UNIX operating system which runs on a large variety of computers. These tools are among the first attempts at automating some of the SCM chores.

SCCS is essentially a library system that forces the members of a programming team to coordinate updating of software module source code, documents or test data. A programmer can withdraw a copy of a module that he wishes to update. Until he submits the updated version, other programmers will be blocked by the system if they also try withdrawing the same module for update. Thus problems caused by simultaneous updating are eliminated.

SCCS also has the ability to retain a record of all modifications made to every module or document that it contains. This includes records of what the changes were, why and when they were made and who made them. It can also upon request reconstruct any previous version of a module or report on the differences between any two versions. Different versions can be maintained simultaneously.

The MAKE utility can work with SCCS to perform an automatic building of a complete software system. After the source code of one or more modules of a system have been modified, the programmer can invoke MAKE. The utility will then analyse the dependency relationships of all modules in the system and their last update times, and take steps to ensure that all modules that depend upon the modified modules will be appropriately updated (eg. by compiling, linking, etc.). The dependency relationships must be provided by other means, usually manually. Since most changes generally do not require the entire system to be recompiled, this tool saves a great deal of recompilation effort.

6.2 Unix RCS

RCS (Revision Control System) is a readily available public domain program that also runs on UNIX computers. It is similar in capabilities to SCCS and is preferred over SCCS at many installations mostly for reasons of performance, and added convenience and flexibility.

6.3 DEC CMS/MMS

The Code Management System (CMS) and Module Management System (MMS), from Digital, are tools that run on VAX computers under the VMS operating system.

CMS is similar to SCCS but provides added features in the areas of security and support for parallel streams of development.

MMS will build an entire software system in the same way that MAKE does. It has the shortcoming however that, unlike the other tools reviewed in this report, it will not support software that is targeted to non-VAX computers.

6.4 Oregon Software SourceTools

SourceTools is a package of software available from Oregon Software which performs the functions of the traditional SCCS and MAKE. It is based on Oregon Software's Pascal and will run on any of the many systems that support that language (eg. DEC PDP-11, DEC VAX, UNIX, PC-DOS).

6.5 Softool CCC

Softool's Change and Configuration Control (CCC) environment is a sophisticated and comprehensive SCM tool that runs on a variety of computers. It has capabilities that go beyond just the management of source code. CCC can:

- Manage the generation and documentation of changes
- Automatically maintain different configurations of software and/or documents

- Improve visibility and streamline control over all activities and procedures during the entire life cycle of a project
- Automate program builds
- Automate procedures of any number of CCC commands that would otherwise be entered manually
- Coordinate the activities of the personnel involved in a project
- Manage all information and processes involved in change and configuration control

CCC can automate the procedures associated with many operations. For example, it can handle the routing of deficiency reports or change requests, automatically notifying responsible personnel, and generate and disseminate reports based on a variety of criteria.

7. STANDARDS AND PRACTICES RECOMMENDATION FOR
THE CANADIAN NUCLEAR POWER INDUSTRY

This report has surveyed the major recognized SCM standards. The overall impression of these standards is that they provide good criteria for developing a software configuration management plan. However they fall short of providing concrete guidance for the practical implementation of SCM procedures especially in areas where SCM claims promise for great benefits. For example, none of the standards adequately address the handling of distributed management or multi-variant software.

When we look at the available automated SCM tools, we find that they provide support that complements the coverage of the SCM standards. This is because the tools originated from a need to address immediate practical problems encountered in everyday software development whereas the standards deliberately avoid implementation details that would narrow their scope of usefulness.

It should also be noted that, because of the loose nature of the SCM standards, SCM plans that comply with them can vary in effectiveness over a wide range. The responsibility for the attainment of SCM benefits rests with the developer of the SCM plan and all those who work under its direction.

It is not possible in this report to provide exact recommendations on how to implement an SCM program within the power utilities or AECL since there are special internal needs of these organizations that must be addressed by any such program. This section will attempt instead to give recommendations on the criteria against which an SCM program should be structured and suggest some features that it should incorporate.

The recommendations are:

1. Adopt standard SCM

Each organization responsible for the production or maintenance of nuclear power station critical software should establish an SCM program based on a recognized standard. Whereas existing station instructions provide control over the process of installing software in operational computers, the adopted SCM should cover all software life cycle phases from design to installation. The practices should be applied to modification of existing software as well as to initial production. Finally, software used for the development and support of the operational software should also be managed.

10 August 1987

2. Use IEEE Std 828
SCM plans should conform to IEEE Std 828. This standard is comprehensive and does not have the drawback of the military standards due to their assumption of a particular software life cycle concept. DOD-STD-2167 and MIL-STD-483 should be used as reference material for expansion of particular topics.
3. Co-ordinate SCM and SQA plans
Since software QA and CM programs are intertwined, their respective plans should be coordinated to ensure that all facets are identified and controls are established.
4. Establish CCBs
A single configuration control board or a hierarchy of CCBs, as deemed appropriate, should be formed at each generating station to enforce formal control over software changes. Critical operational software baselines should be under the control of high level management of the organization, whereas lower levels of authority should be applied to less critical baselines. The IEEE SCM guide (ref [10]) should be consulted on this issue.
5. Distributed management
To exploit the opportunity to maintain a software item that is used at multiple generating stations as a single item rather than multiple variants, one of two possible courses of action should be taken.
 - a. Allow all stations to make required changes locally. These changes would then be submitted to a central joint CCB with representatives from all stations. Approved changes would then be combined into one central baseline which would be available for release to all stations.
 - b. Each change approved by the local station CCB would be forwarded to a central joint CCB. If approved at that level, the changes would be incorporated into a single baseline for release to all stations.

Communications channels should be opened for the dissemination of deficiency and change information among stations.
6. Use automation
The current software development computers should be provided with at least a basic SCM automation capability as defined in the IEEE SCM guide (ref [10]). Commercially available SCM tools are available for many of the development system computers, including the IBM PC.

10 August 1987

As an alternative, consideration should be given to installing a central automated source code repository located with the configuration control authority (eg. at each station). It would assist in the processing of change documentation, control of software baselines and associated documentation, configuration status accounting, and other tasks. (A possible repository system might be a stand-alone computer, running SCM software like Softool CCC, linked to the software development systems by communications links for interchanging documentation and source code modules.)

7. Multi-variant software

The problems of maintaining different versions of the same generic software item in use within the same generating station or at different stations should be addressed using automated source code management tools. These safely and efficiently manage such configurations by guaranteeing that the components that are supposed to be identical in all variants remain identical.

APPENDIX A STANDARDS BODY ADDRESSES AND DATA

ANSI/IEEE Std 828

Software Engineering Technical Committee of the
IEEE Computer Society
The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA

ANSI/IEEE Std 1033-1985

Nuclear Power Engineering Committee of the
IEEE Power Engineering Society
The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA

ANSI/IEEE-ANS-7-4.3.2-1982

American Nuclear Society and
The Institute of Electrical and Electronics Engineers, Inc.

American Nuclear Society
555 North Kensington Avenue
La Grange Park, Illinois 60525 USA

British Standard 6488: 1984

Publication Manager
British Standards Institution
Linford Wood,
Milton Keynes
MK14 6LE

DOD-STD-2167

Commander
Space and Naval Warfare Systems Command
Standardization and Data Management Branch
Department of the Navy
Washington, DC 20363-5100
Attn: SPAWAR 8111

MIL-STD-483

Air Force Systems Command, AFSC (SDMA),
Andrews Air Force Base,
Washington, DC 20331

10 August 1987

APPENDIX B ACTUAL USE OF SCM

B.1 Transport Canada Air Traffic Control Systems

Automated systems have been in wide use for a number of years in the support of air traffic control services in Transport Canada. A software configuration control plan was set up about 1981 at the time when the major radar processing system (JETS) and flight data processing system (NFDPS) were being installed in the seven area control centers across the country. At present, the plan covers six different types of systems, four of which exist at multiple sites.

The plan is implemented as a series of departmental directives which cover topics such as:

- designation of systems subject to configuration control,
- headquarters and regional responsibilities,
- delegation of approval authority at headquarters,
- procedures for originating unsatisfactory condition reports (UCRs), and change proposals (CPs),
- headquarters procedures for processing UCRs and CPs,
- procedures for implementation of system modifications,
- guidelines for classifying UCRs.

All software is managed and maintained by one organization at headquarters in Ottawa where a permanent project team is dedicated to maintaining the software for each major system. No facilities for changing software exist at the operational sites.

A uniform method of reporting faults is implemented across all systems. The originator (at a field site or at headquarters) fills in a UCR form and submits it to the headquarters project office. The identifier on a UCR designates the affected system, the originating site and the site's UCR sequence number.

Requests and proposals for changes not related to faults are submitted as a CP (Change Proposal) in the same way as UCRs.

UCRs and CPs received at headquarters are processed through an organization that has several levels of pre-delegated authority for approving changes. The organization has the following entities:

- System Office Review Team (SORT)
Consists of system project team members and other specialists as required.
- Office of Primary Responsibility (OPI)

10 August 1987

One or both of two department branches depending on the nature of the UCR/CP: Air Traffic Services (ATS) (the operations branch), or Facility Engineering and System Development (FESD) (the technical branch).

- System Configuration Control Committee (SCCC)
Superintendents of the designated ATS and FESD system offices.
- National ATS Systems Configuration Control Board (NASCCB)
Chief or director levels of management based on authority delegation criteria.

The SORT periodically meets on a convenient schedule to perform the following functions:

- review interim solutions to urgent UCRs
- classify and prioritize UCRs/CPs
- assign OPI for UCR/CP processing
- review status of UCRs/CPs in progress
- make recommendations to SCCC for some CPs

The SORT, in practice, authorizes most of the routine repair changes without involving higher authorities. Changes with important consequences are delegated for higher level approval as required.

UCRs classified as urgent are dealt with immediately by the software maintenance staff with the goal of producing an easily installable correction or patch. Very urgent trouble reports can be phoned to the support office on a 24 hour basis. A SORT meeting will review the interim solution at the earliest convenience with the same procedures as for non-urgent UCRs.

All software and documentation is maintained on-line on the various development systems used for software maintenance. No extensive automated tools are available. All program support library functions are carried out with only minimal automation.

A modified software release is delivered to a site with an accompanying release notes document. The site staff are encouraged to certify the new release by running the software through a suitable proof of performance (POP) test. In some cases POP test software is also delivered with the operational software.

B.2 Department of National Defence: DMCS-8

The Fleet Software Support Center in DND is in the process of setting up procedures and tools for complete management of several software systems.

Their procedures are based on DOD-STD-2167. They did not perform an extensive survey of standards to arrive at this decision, although they did review MIL-STD-483 and found that its high level of detail allowed insufficient flexibility for developing procedures tailored to their needs.

In selecting automated tools to support their SCM procedures, they evaluated only Softool's CCC and DEC's CMS/MMS. They selected CCC because it conformed with their software development environment (VAX/VMS with HOST-16) and it appeared to meet all their requirements. Their main requirements were:

- 1) a high level of automation,
- 2) ease of learning,
- 3) ease of use, and
- 4) a high degree of protection and control.

CMS/MMS was found to not have the required protection features.

The establishment of the system is not yet complete so no practical experience has been obtained.

10 August 1987

B.3 Bell Labs, Electronic Switching Systems

Bell Labs has pioneered a long history of software configuration management in the development of numerous programmable electronic telephone switching systems. This appendix gives highlights of the evolution of SCM procedures and tools that has come about during this development.

Bell Lab's SCM practices are more strongly oriented to the use of automated support tools rather than to management issues. This is likely due to the fact that all software was developed within the company rather than being procured through contract.

B.3.1 No. 4 ESS

The Number 4 Electronic Switching System (ESS) was developed during the mid 1970s (ref [12]). A very large amount of software was produced (about 1,000,000 words of object code) by a staff of 200 program designers. At the onset of the project it was deemed essential to have an all inclusive automated facility for programmers and administrators.

The Program Administration System (PAS) was created to provide for creation, modification, preservation, and administration of the software. PAS was implemented on several mainframe computers providing interactive and batch processing of requests from the community of users. The system maintained a database of application software accessible by users. Special privileges were granted to an administrator who exercised control over the programmer's privileges and access rights to the database. PAS contained facilities for development of changes to established software such as special assemblers and loaders for producing and testing incremental loads of target software. The administrator was able to deny programmer access to frozen versions of software that were under strict change control.

Although PAS was a significant system for its era, most of its capabilities for storage and protection exist in today's general purpose multi-user operating systems.

B.3.2 TSPS No. 1

Maintenance of the Traffic Service Position System (TSPS), No. 1, (ref [14]) software required managing multiple variants of the software for different field sites. The variants were called generic releases. Minor releases provided minor enhancements to a generic. It was decided to maintain as many versions of a software module as there were active generics. Rather than storing multiple copies of a module, conditional assembly statements were added to the source code to support the different feature levels.

The management practices required programmers to work on the automated system under one of two administrative modes: the "development" mode and the "frozen" mode.

Development mode was used for the generation of new features. It gave programmers as much freedom as possible. A special Program Administration Group (PAG) would merge changes made by the development teams.

The frozen mode was applied to generics that had been issued to the field. It was designed to give the maximum amount of stability by providing a highly controlled and documented change environment through which programmers made corrections. Trouble Reports (TRs) were used to document a problem or a need for improvement.

The frozen mode had the following restrictions:

1. No change to a module could cause its object code size to change.
2. All programmer-generated changes must be independently tested and approved by a generic test team.
3. The application programmer must generate a written Correction Report (CR) documenting changes made in response to a TR.
4. Not only must the change be independently tested, but the TR, CR and software modifications must be approved by a generic software change review committee prior to the changes being incorporated in the next release of the frozen generic.

The TSPS No.1 had the following change implementation flow:

1. The TR was received by the application programmer who would be making the software change.
2. The application programmer would generate an object code

overwrite (using a special assembler) to fix the stated problem and test the fix by temporarily overwriting the generic load image in the laboratory system.

3. Satisfied with the results, the application programmer would generate the corresponding CR and source code edits which produced object code identical to the overwrite.
4. The TR, CR, overwrite and source edits would then be submitted to the generic software change review committee for approval. If rejected, go back to step 2.
5. If approved, the TR, CR, overwrite and source edits would be submitted to the generic system test team for independent overwrite testing. If rejected, back to step 2.
6. If tests were passed, the TR, CR, overwrite and source edits would be submitted to the PAG. The source edits would be incorporated with other possible edits for the module(s) involved. The TR and CR would be filed and the overwrite saved.

This procedure would be followed for each TR requiring a software change. When a new release of the frozen generic would be required, the PAG would reassemble all modified modules and regenerate the generic load. The load was required to exactly match the overwrite load image in the laboratory system before it would be released to Western Electric for distribution.

B.3.3 TSPS No. 1B

In the development of Traffic Service Position System (TSPS), No. 1B, (ref [9]) an SCM tool was produced called Change Management System (CMS). It provides the ability to keep track of the current version of the official program source, isolate test versions of programs, and eventually introduce either new code or corrections into the official version of the operational software. It provides not just isolation of changes in different stages of development but also independence of one generic from another.

In CMS all development activity is tied to the notion of a modification request (MR). Correction reports (CRs) are submitted by developers. After testing, these are approved by a change review committee.

B.3.4 No. 5 ESS

The number 5 ESS is one of Bell Lab's most recent electronic switching systems (ref [2]). The software development and maintenance for this system is supported by a collection of computer aids known as the System Development Environment (SDE). It provides control and tracking of all source and object code for each development version of a system as well as for each officially released version.

SDE runs on a network of computers and includes CMS (described above) and another major component called Software Change Administration (SCANS). SCANS receives updates for field software from the development portions of SDE. It packages and transmits these updates to the field.

APPENDIX C BIBLIOGRAPHY

- [1] W.A. Babich, "Software Configuration Management", Addison-Wesley, Reading, Massachusetts, 1986
- [2] R.G. Basinger, J.A. Herndon, B. Kaskey, J.A. Lindner, J.M. Milner, "The SESS Switching System: System Development Environment", AT&T Technical Journal, Vol. 64, No. 6, Jul-Aug 1985
- [3] R. Bazelmans, "Evolution of Configuration Management", ACM Software Engineering Notes, Vol 10, No. 5, Oct 1985
- [4] E. Bersoff, V. Henderson, S. Siegel, "Software Configuration Management", Englewood Cliffs, NJ: Prentice-Hall, 1980
- [5] Bersoff, E.H., "Elements of Software Configuration Management", IEEE Transactions on Software Engineering, 10(1), p79-87 (1984)
- [6] W.Bryan, C.Chadbourn, S.Siegel, "Tutorial: Software Configuration Management", IEEE Computer Society Press, 1980
- [7] J.K. Buckle, "Software Configuration Management", The Macmillan Press, London, 1982
- [8] Defense Logistics Agency, Handbook 334 for MIL-S-52779A, DLAH 8250.1, May 1981
- [9] T.G. Hack, T. Huang, L.C. Stecher, "Traffic Service Position No. 1B: Software Development System", Bell System Technical Journal, Vol 62, No. 3, March 1983.
- [10] IEEE Computer Society, "Guide to Software Configuration Management", (Only available as unapproved draft to this date. Official publication by IEEE expected in October 1987.)
- [11] K. Ives, "Computer Software Quality Assurance", AECB Research Report INFO-0201, June 1986
- [12] P.S. McCabe, J.B. Otto, S. Roy, G.A. Sellers Jr., K.W. Zweifwyl, "No. 4ESS: Program Administration, Test, and Evaluation", Bell System Technical Journal, Vol 56, No. 7, Sept. 1977

- [13] J.A. McCall, M.A. Herdon, and W.M. Osborne,
"Software Maintenance Management",
National Bureau of Standards, NBS Special Publication
500-129, Oct 1985
- [14] J.J. Stanaway, Jr., J.J. Victor, R.J. Welsch, "Traffic
Service Position System No.1: Software Development Tools",
Bell System Technical Journal, Vol. 58, No. 6, Jul-Aug 1979
- [15] W.F. Tichy, "RCS - A System for Version Control",
Software Practice and Experience, Vol. 15, No. 7, July 1985